Assignment #11

Jingye Wang

December 2, 2016

Ch24.1

Download the data in the folder dogs and fit some other models, for example using as a predictor the result from the previous trial, or the previous two trials, rather than the total number of shocks and avoidances.

```
options(stringAsFactors = F, max.print = 10000)
setwd("~/Dropbox/WUSTL third/Multilevel Modeling for Quantitative Research/assignment/1
1")
.libPaths("/Library/Frameworks/R.framework/Versions/3.3/Resources/library")
require(R2jags)
require(rjags)
require(dplyr)
require(magrittr)
dogs_data <- read.table('dogs2.txt', row.names = 1)
y <- dogs_data
n_dogs <- nrow(y)
n_trials <- ncol(y)
```

Part A

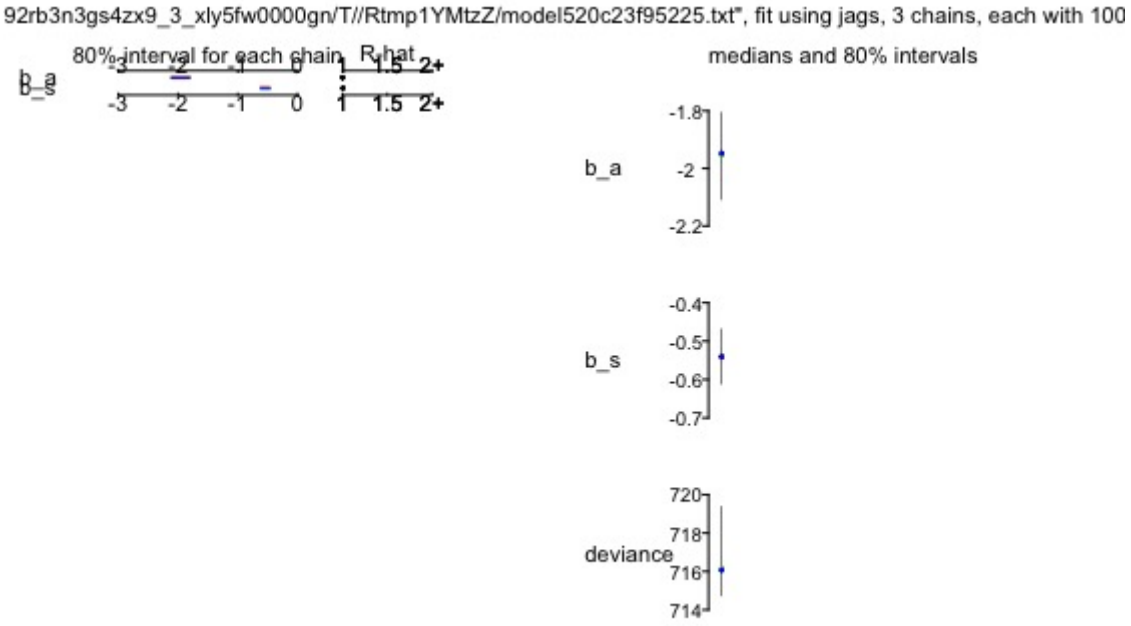
Fit this model, as usual building up from simpler versions (first a single- level model, then varying intercepts, then varying slopes, then adding other predictors as appropriate). Plot the data and fitted model to make sure that your model makes sense.

Single Level Model

```
dogs_1 <- function() {
 for (j in 1:n_dogs) {
   pre1_s[j, 1] <- 0
   pre1_a[j, 1] <- 0
   for (t in 2:n_trials) {
     pre1_s[j, t] <- y[j, t-1]
     pre1_a[j, t] <- 1 - y[j, t-1]
   }
   for (t in 1:n_trials) {
     y[j, t] ~ dbin(p[j, t], 1)
     log(p[j, t]) <- b_s * pre1_s[j, t] + b_a * pre1_a[j, t]
   }
 }
 b_s <- -b_s_neg
 b_a <- -b_a_neg
 b_s_neg ~ dlnorm(mu_b_s, tau_b_s)
 b_a_neg ~ dlnorm(mu_b_a, tau_b_a)
 tau_b_s <- pow(sigma_b_s, -2)
 tau_b_a <- pow(sigma_b_a, -2)

 mu_b_s ~ dnorm(0, .0001)
 mu_b_a ~ dnorm(0, .0001)
 sigma_b_s ~ dunif(0, 100)
 sigma_b_a ~ dunif(0, 100)
}

data_1 <- list("y", "n_dogs", "n_trials")
inits_1 <- function (){
 list(b_s_neg=rlnorm(1),
      b_a_neg=rlnorm(1),
      mu_b_s= rnorm(1),
      mu_b_a=rnorm(1),
      sigma_b_s=dunif(1, 0, 100),
      sigma_b_a=dunif(1, 0, 100))
}
params_1 <- c ('b_s', 'b_a')

dogs_1_jags <- jags(data_1, inits_1, params_1, n.chains = 3, n.iter = 1000, dogs_1)
plot(dogs_1_jags)
dogs_1_jags$BUGSoutput
```

Output

```
Inference for Bugs model at "/var/folders/ld/tp92rb3n3gs4zx9_3_xly5fw0000gn/T//RtmplYMtz
Z/model520c23f95225.txt", fit using jags,
 3 chains, each with 1000 iterations (first 500 discarded)
 n.sims = 1500 iterations saved
          mean  sd  2.5%   25%   50%   75% 97.5% Rhat n.eff
b_a       -2.0 0.1  -2.2  -2.0  -1.9  -1.9  -1.7    1  1500
b_s       -0.5 0.1  -0.7  -0.6  -0.5  -0.5  -0.4    1   580
deviance 716.7 2.1 714.6 715.1 716.1 717.6 722.0    1  1500

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

DIC info (using the rule, pD = var(deviance)/2)
pD = 2.1 and DIC = 718.8
DIC is an estimate of expected predictive error (lower deviance is better).
```



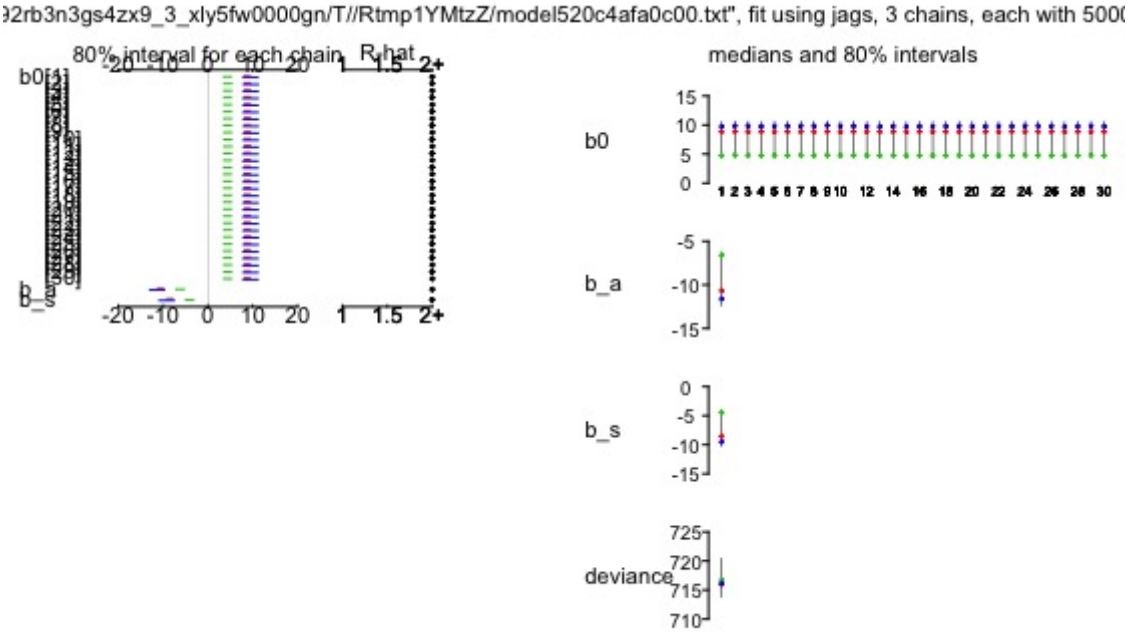
Based on this single lever model, both of the shock (slope = -0.5) and the avoid (slope = -2.0) of the previous trials decrease the possibility of shock and the avoid plays a more important role compared to the shock.

Varying Intercept Model

```
dogs_2 <- function() {
 for (j in 1:n_dogs) {
   pre1_s[j, 1] <- 0
   pre1_a[j, 1] <- 0
   for (t in 2:n_trials) {
     pre1_s[j, t] <- y[j, t-1]
     pre1_a[j, t] <- 1 - y[j, t-1]
   }
   for (t in 1:n_trials) {
     y[j, t] ~ dbin(p[j, t], 1)
     logit(p[j, t]) <- b_s * pre1_s[j, t] +  b_a * pre1_a[j, t] + b0[j]
   }
   b0[j] ~ dnorm(mu_b0, tau_b0)
 }
 mu_b0 ~ dnorm(0, .0001)
 tau_b0 <- pow(sigma_b0, -2)
 sigma_b0 ~ dunif(0, 100)

 b_s ~ dnorm(mu_b_s, tau_b_s)
 b_a ~ dnorm(mu_b_a, tau_b_a)
 tau_b_s <- pow(sigma_b_s, -2)
 tau_b_a <- pow(sigma_b_a, -2)
 mu_b_s ~ dnorm(0, .0001)
 mu_b_a ~ dnorm(0, .0001)
 sigma_b_s ~ dunif(0, 100)
 sigma_b_a ~ dunif(0, 100)
}

data_2 <- list("y", "n_dogs", "n_trials")
inits_2 <- function () {
 list(b_s=rnorm(1),
      b_a=rnorm(1),
      mu_b_s= rnorm(1),
      mu_b_a=rnorm(1),
      sigma_b_s=dunif(1, 0, 100),
      sigma_b_a=dunif(1, 0, 100),
      b0=rnorm(n_dogs),
      mu_b0=rnorm(1),
      sigma_b0=runif(1, 0, 100))
}
params_2 <- c ('b_s', 'b_a', 'b0')

dogs_2_jags <- jags(data_2, inits_2, params_2, n.chains = 3, n.iter = 5000, dogs_2)\

plot(dogs_2_jags)
dogs_2_jags$BUGSoutput
```

Output

```
Inference for Bugs model at "/var/folders/ld/tp92rb3n3gs4zx9_3_xly5fw0000gn/T//Rtmp1YMtz
Z/model520c4afa0c00.txt", fit using jags,
 3 chains, each with 5000 iterations (first 2500 discarded), n.thin = 2
 n.sims = 3750 iterations saved
          mean  sd  2.5%   25%   50%   75% 97.5% Rhat n.eff
b0[1]      7.7 2.4   3.4   5.1   8.5   9.3  11.2  5.1     3
b0[2]      7.7 2.4   3.4   5.1   8.6   9.3  11.3  5.1     3
b0[3]      7.7 2.4   3.4   5.1   8.5   9.3  11.3  5.1     3
b0[4]      7.7 2.4   3.4   5.1   8.5   9.3  11.2  5.1     3
b0[5]      7.7 2.4   3.4   5.1   8.5   9.3  11.2  5.1     3
...
b0[25]     7.7 2.4   3.3   5.1   8.5   9.3  11.2  5.1     3
b0[26]     7.7 2.4   3.3   5.1   8.5   9.3  11.2  5.1     3
b0[27]     7.7 2.4   3.3   5.1   8.5   9.3  11.2  5.1     3
b0[28]     7.7 2.4   3.4   5.1   8.5   9.3  11.2  5.1     3
b0[29]     7.7 2.4   3.4   5.1   8.5   9.3  11.2  5.1     3
b0[30]     7.7 2.4   3.3   5.1   8.5   9.3  11.2  5.1     3
b_a       -9.5 2.4 -13.0 -11.1 -10.3  -6.9  -5.2  4.7     3
b_s       -7.4 2.4 -10.9  -9.0  -8.2  -4.8  -3.1  4.7     3
deviance 716.8 2.9 711.9 715.0 716.3 718.1 723.7  1.0   170

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

DIC info (using the rule, pD = var(deviance)/2)
pD = 4.1 and DIC = 720.8
DIC is an estimate of expected predictive error (lower deviance is better).
```



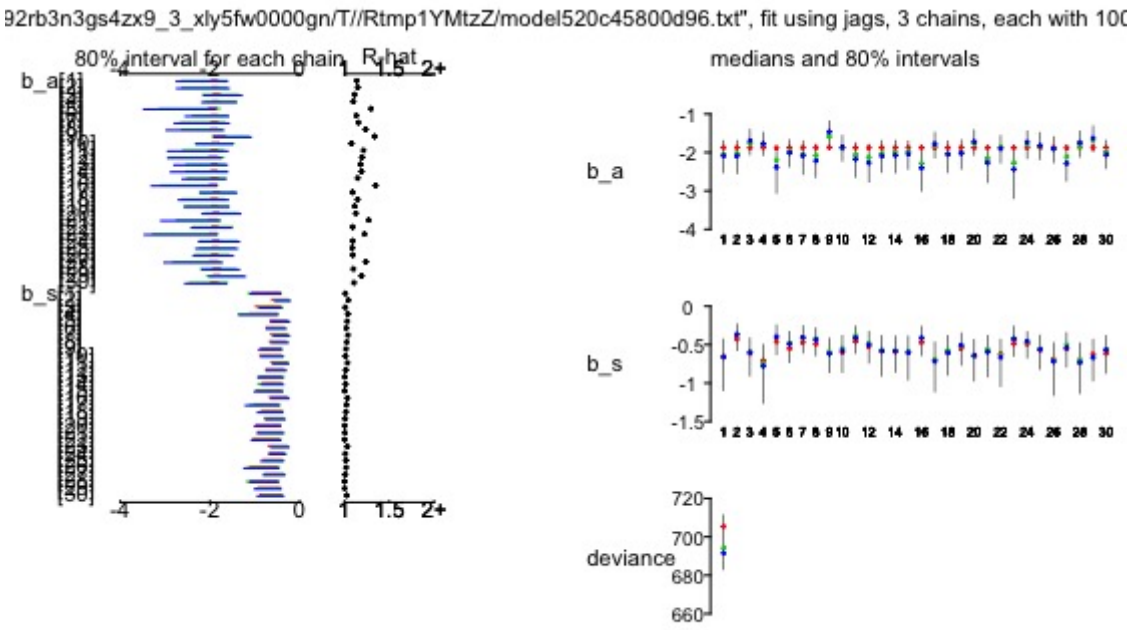
Based on this varying intercept model, both of the shock (slope = -7.4) and the avoid (slope = -9.5) of the previous trials decrease the possibility of shock and the avoid plays a more important role compared to the shock. The varying intercepts doesn't change much among different dogs.

Varying Slopes Model

```
dogs_3 <- function() {
 for (j in 1:n_dogs) {
   pre1_s[j, 1] <- 0
   pre1_a[j, 1] <- 0
   for (t in 2:n_trials) {
     pre1_s[j, t] <- y[j, t-1]
     pre1_a[j, t] <- 1 - y[j, t-1]
   }
   for (t in 1:n_trials) {
     y[j, t] ~ dbin(p[j, t], 1)
     log(p[j, t]) <- b_s[j] * pre1_s[j, t] + b_a[j] * pre1_a[j, t]
   }
   b_s[j] <- -b_s_neg[j]
   b_a[j] <- -b_a_neg[j]
   b_s_neg[j] ~ dlnorm(mu_b_s, tau_b_s)
   b_a_neg[j] ~ dlnorm(mu_b_a, tau_b_a)
 }
 tau_b_s <- pow(sigma_b_s, -2)
 tau_b_a <- pow(sigma_b_a, -2)
 mu_b_s ~ dnorm(0, .0001)
 mu_b_a ~ dnorm(0, .0001)
 sigma_b_s ~ dunif(0, 100)
 sigma_b_a ~ dunif(0, 100)
}

data_3 <- list("y", "n_dogs", "n_trials")
inits_3 <- function (){
 list(b_s_neg=rlnorm(n_dogs),
      b_a_neg=rlnorm(n_dogs),
      mu_b_s= rnorm(1),
      mu_b_a=rnorm(1),
      sigma_b_s=dunif(1, 0, 100),
      sigma_b_a=dunif(1, 0, 100))
}
params_3 <- c ('b_s', 'b_a')

dogs_3_jags <- jags(data_3, inits_3, params_3, n.chains = 3, n.iter = 1000, dogs_3)

plot(dogs_3_jags)
dogs_3_jags$BUGSoutput
```

Output

```
Inference for Bugs model at "/var/folders/ld/tp92rb3n3gs4zx9_3_xly5fw0000gn/T//RtmplYMtz
Z/model520c45800d96.txt", fit using jags,
 3 chains, each with 1000 iterations (first 500 discarded)
 n.sims = 1500 iterations saved
          mean   sd  2.5%   25%   50%   75% 97.5% Rhat n.eff
b_a[1]    -2.0  0.4  -3.1  -2.2  -1.9  -1.9  -1.5  1.1    27
b_a[2]    -2.0  0.4  -3.0  -2.2  -1.9  -1.9  -1.4  1.1    27
b_a[3]    -1.8  0.3  -2.4  -1.9  -1.8  -1.6  -1.1  1.1    44
b_a[4]    -1.8  0.2  -2.3  -1.9  -1.9  -1.7  -1.3  1.1    62
b_a[5]    -2.3  0.6  -4.0  -2.5  -2.1  -1.9  -1.7  1.3    11
...
b_a[25]   -1.9  0.3  -2.5  -2.0  -1.9  -1.7  -1.3  1.1   370
b_a[26]   -1.9  0.3  -2.6  -2.0  -1.9  -1.8  -1.4  1.1   240
b_a[27]   -2.2  0.5  -3.3  -2.3  -2.0  -1.9  -1.6  1.2    14
b_a[28]   -1.8  0.3  -2.4  -1.9  -1.9  -1.7  -1.2  1.1    86
b_a[29]   -1.7  0.3  -2.2  -1.9  -1.8  -1.5  -1.1  1.2    17
b_a[30]   -2.0  0.3  -2.8  -2.2  -1.9  -1.8  -1.5  1.1    41
b_s[1]    -0.7  0.3  -1.5  -0.9  -0.7  -0.5  -0.3  1.0   690
b_s[2]    -0.4  0.1  -0.7  -0.5  -0.4  -0.3  -0.2  1.0    53
b_s[3]    -0.6  0.2  -1.1  -0.7  -0.6  -0.5  -0.3  1.0   260
b_s[4]    -0.8  0.4  -1.8  -1.0  -0.7  -0.6  -0.4  1.0   140
b_s[5]    -0.4  0.2  -0.8  -0.5  -0.4  -0.3  -0.2  1.0    84
...
b_s[25]   -0.6  0.2  -1.0  -0.7  -0.6  -0.5  -0.3  1.0   230
b_s[26]   -0.8  0.3  -1.6  -0.9  -0.7  -0.6  -0.4  1.0   310
b_s[27]   -0.6  0.2  -1.0  -0.7  -0.5  -0.4  -0.3  1.0   460
b_s[28]   -0.8  0.3  -1.5  -0.9  -0.7  -0.6  -0.4  1.0  1500
b_s[29]   -0.7  0.2  -1.2  -0.8  -0.6  -0.5  -0.3  1.0   890
b_s[30]   -0.6  0.2  -1.2  -0.7  -0.6  -0.5  -0.3  1.0   440
deviance 697.1 10.8 676.1 689.7 697.2 704.9 716.5  1.5     8

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

DIC info (using the rule, pD = var(deviance)/2)
pD = 41.2 and DIC = 738.3
DIC is an estimate of expected predictive error (lower deviance is better).
```



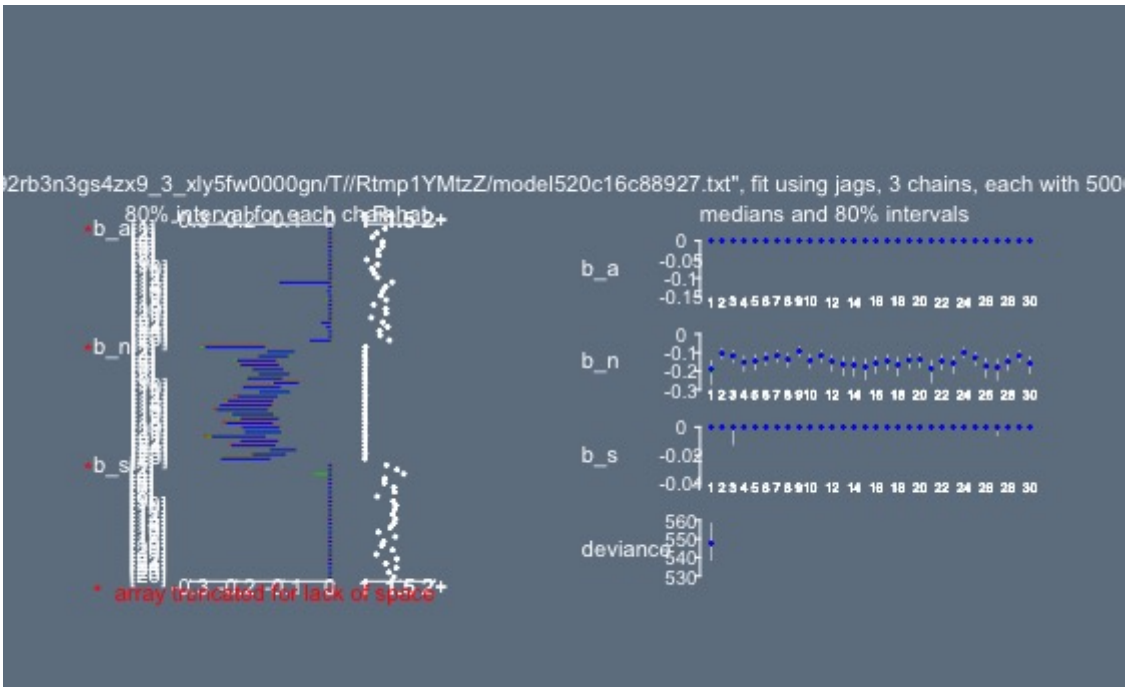
Based on this varying slope model, both of the shock (slope = -0.4 ~ -0.8) and the avoid (slope = -1.8 ~ -2.3) of the previous trials decrease the possibility of shock and the avoid plays a more important role compared to the shock.

Additional Predictor Model

```
# Model_4 (additional predictor)
dogs_4 <- function() {
 for (j in 1:n_dogs) {
   pre1_s[j, 1] <- 0
   pre1_a[j, 1] <- 0
   pre_num_trials[j, 1] <- 0
   for (t in 2:n_trials) {
     pre1_s[j, t] <- y[j, t-1]
     pre1_a[j, t] <- 1 - y[j, t-1]
     pre_num_trials[j, t] <- t - 1
   }
   for (t in 1:n_trials) {
     y[j, t] ~ dbin(p[j, t], 1)
     log(p[j, t]) <- b_s[j] * pre1_s[j, t] + b_a[j] * pre1_a[j, t] + b_n[j] * pre_num_tr
ials[j, t]
   }
   b_n[j] <- -b_n_neg[j]
   b_s[j] <- -b_s_neg[j]
   b_a[j] <- -b_a_neg[j]
   b_n_neg[j] ~ dlnorm(mu_b_n, tau_b_n)
   b_s_neg[j] ~ dlnorm(mu_b_s, tau_b_s)
   b_a_neg[j] ~ dlnorm(mu_b_a, tau_b_a)
 }
 mu_b_n ~ dnorm(0, .0001)
 mu_b_s ~ dnorm(0, .0001)
 mu_b_a ~ dnorm(0, .0001)
 tau_b_n <- pow(sigma_b_n, -2)
 tau_b_s <- pow(sigma_b_s, -2)
 tau_b_a <- pow(sigma_b_a, -2)
 sigma_b_n ~ dunif(0, 100)
 sigma_b_s ~ dunif(0, 100)
 sigma_b_a ~ dunif(0, 100)
}

data_4 <- list("y", "n_dogs", "n_trials")
inits_4 <- function (){
 list(
   b_n_neg=rlnorm(n_dogs),
   b_s_neg=rlnorm(n_dogs),
   b_a_neg=rlnorm(n_dogs),
   mu_b_n= rnorm(1),
   mu_b_s=rnorm(1),
   mu_b_a=rnorm(1),
   sigma_b_n=dunif(1, 0, 100),
   sigma_b_s=dunif(1, 0, 100),
   sigma_b_a=dunif(1, 0, 100))
}
params_4 <- c ('b_s', 'b_a', 'b_n')

dogs_4_jags <- jags(data_4, inits_4, params_4, n.chains = 3, n.iter = 5000, dogs_4)

plot(dogs_4_jags)
dogs_4_jags$BUGSoutput
```

Output

```
Inference for Bugs model at "/var/folders/ld/tp92rb3n3gs4zx9_3_xly5fw0000gn/T//RtmplYMtz
Z/model520c16c88927.txt", fit using jags,
 3 chains, each with 5000 iterations (first 2500 discarded), n.thin = 2
 n.sims = 3750 iterations saved
          mean  sd  2.5%   25%   50%   75% 97.5% Rhat n.eff
b_a[1]     0.0 0.0   0.0   0.0   0.0   0.0   0.0  1.3    73
b_a[2]     0.0 0.0   0.0   0.0   0.0   0.0   0.0  1.2  1200
b_a[3]     0.0 0.0   0.0   0.0   0.0   0.0   0.0  1.1   550
b_a[4]     0.0 0.0   0.0   0.0   0.0   0.0   0.0  1.3   220
b_a[5]     0.0 0.0   0.0   0.0   0.0   0.0   0.0  1.2  1400
...
b_a[25]    0.0 0.0   0.0   0.0   0.0   0.0   0.0  1.2   220
b_a[26]    0.0 0.0  -0.1   0.0   0.0   0.0   0.0  1.3    31
b_a[27]    0.0 0.0   0.0   0.0   0.0   0.0   0.0  1.3    97
b_a[28]    0.0 0.0   0.0   0.0   0.0   0.0   0.0  1.2    70
b_a[29]    0.0 0.0   0.0   0.0   0.0   0.0   0.0  1.3   180
b_a[30]    0.0 0.0   0.0   0.0   0.0   0.0   0.0  1.2   250
b_n[1]    -0.2 0.1  -0.3  -0.2  -0.2  -0.2  -0.1  1.0   750
b_n[2]    -0.1 0.0  -0.2  -0.1  -0.1  -0.1  -0.1  1.0  1700
b_n[3]    -0.1 0.0  -0.2  -0.1  -0.1  -0.1  -0.1  1.0  3800
b_n[4]    -0.2 0.0  -0.2  -0.2  -0.2  -0.1  -0.1  1.0  2200
b_n[5]    -0.2 0.0  -0.2  -0.2  -0.1  -0.1  -0.1  1.0  3800
...
b_n[25]   -0.1 0.0  -0.2  -0.1  -0.1  -0.1  -0.1  1.0   530
b_n[26]   -0.2 0.0  -0.3  -0.2  -0.2  -0.1  -0.1  1.0  3800
b_n[27]   -0.2 0.0  -0.3  -0.2  -0.2  -0.2  -0.1  1.0   460
b_n[28]   -0.2 0.0  -0.2  -0.2  -0.1  -0.1  -0.1  1.0  3800
b_n[29]   -0.1 0.0  -0.2  -0.1  -0.1  -0.1  -0.1  1.0  3800
b_n[30]   -0.2 0.0  -0.3  -0.2  -0.2  -0.1  -0.1  1.0  3800
b_s[1]     0.0 0.0   0.0   0.0   0.0   0.0   0.0  1.3    52
b_s[2]     0.0 0.0   0.0   0.0   0.0   0.0   0.0  1.5    13
b_s[3]     0.0 0.0   0.0   0.0   0.0   0.0   0.0  1.6    10
b_s[4]     0.0 0.0   0.0   0.0   0.0   0.0   0.0  1.3    38
b_s[5]     0.0 0.0   0.0   0.0   0.0   0.0   0.0  1.3    16
...
b_s[25]    0.0 0.0   0.0   0.0   0.0   0.0   0.0  1.3    18
b_s[26]    0.0 0.0   0.0   0.0   0.0   0.0   0.0  1.4    17
b_s[27]    0.0 0.0   0.0   0.0   0.0   0.0   0.0  1.8     7
b_s[28]    0.0 0.0   0.0   0.0   0.0   0.0   0.0  1.4    16
b_s[29]    0.0 0.0   0.0   0.0   0.0   0.0   0.0  1.4    14
b_s[30]    0.0 0.0   0.0   0.0   0.0   0.0   0.0  1.5    14
deviance 547.9 7.7 534.9 542.4 547.4 552.9 564.4  1.0  2800

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

DIC info (using the rule, pD = var(deviance)/2)
pD = 29.4 and DIC = 577.3
DIC is an estimate of expected predictive error (lower deviance is better).
```



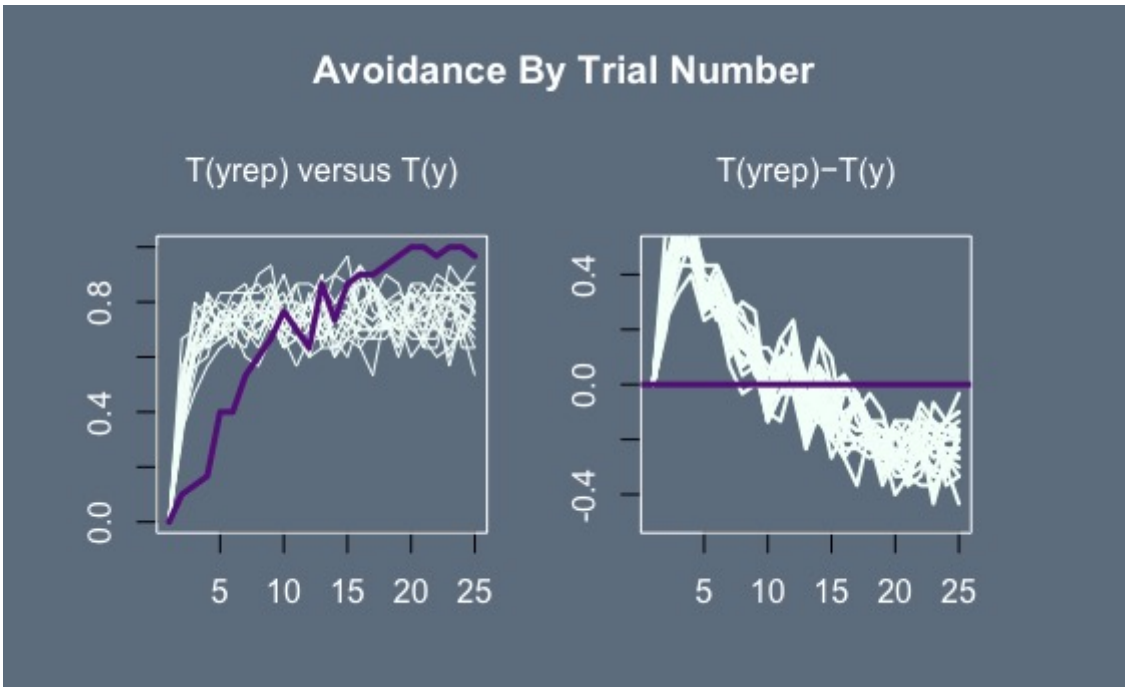
For this additional predictor model, I add number of trials as a third predictor. After adding this predictor, both of the shock (slope = 0) and the avoid (slope = 0) doesn't matter for this model. However, the number of trials (slope = -0.1 ~ -0.2) plays the most important role.

Part B

Use Bugs to simulate replicated datasets from your model, and make various plots to compare the replicated with the actual data.

Simulation

```
b_a <- dogs_3_jags$BUGSoutput$summary[1:30, 1]
b_s <- dogs_3_jags$BUGSoutput$summary[31:60, 1]

n_sims <- 1000
y_rep <- array(NA, c(n_sims, n_dogs, n_trials))
for (j in 1:n_dogs){
 pre1_a_rep <- rep (0, n_sims)
 pre1_s_rep <- rep (0, n_sims)
 for (t in 1:n_trials){
   p_rep <- exp (b_a[j] * pre1_a_rep + b_s[j] * pre1_s_rep)
   y_rep[ , j, t] <- rbinom (n_sims, 1, p_rep)
   pre1_a_rep <- 1 - y_rep[ , j, t]
   pre1_s_rep <- y_rep[ , j, t]
 }
}

par(mfrow=c(1,2),mar=c(2,2,2,2),oma=c(2,2,4,2),col.axis="white",col.main="white",
   col.lab="white",col.sub="white",col="white",bg="slategray")
plot(1:25,1-apply(dogs_data,2,mean),type="n")
rand.sample <- sample(x=1:n_sims,size=20)
for (i in rand.sample) lines(1:25,1-apply(y_rep[i,,],2,mean),col="mintcream")
lines(1:25,1-apply(dogs_data,2,mean),col="darkorchid4",lwd=3,type="l")
mtext(side=3,line=1.25,"T(yrep) versus T(y)")
plot(1:25,seq(-0.5,0.5,length=25),type="n")
for (i in rand.sample)
 lines(1:25,(1-apply(y_rep[i,,],2,mean))-(1-apply(dogs_data,2,mean)),
       col="mintcream",lwd=2,type="l")
abline(h=0,col="darkorchid4",lwd=3)
mtext(side=3,line=1.25,"T(yrep)-T(y)")
title(outer=TRUE,line=2,col="white","Avoidance By Trial Number")
```



I choose to use the varying slope model to do the simulation part. The simulation curves remain much higher than the actual data at the first 10 trials, but lower than the actual data at the last 15 trials, whcih means this model is not appropriate in predicting the results.

Ch24.4

Model checking for ordered categorical regression:

Part A

Do some simulation-based graphical checking for the ordered logistic regression model that you fit in Exercise 17.11 to the data from the storable-voting experiment.

Model Fit

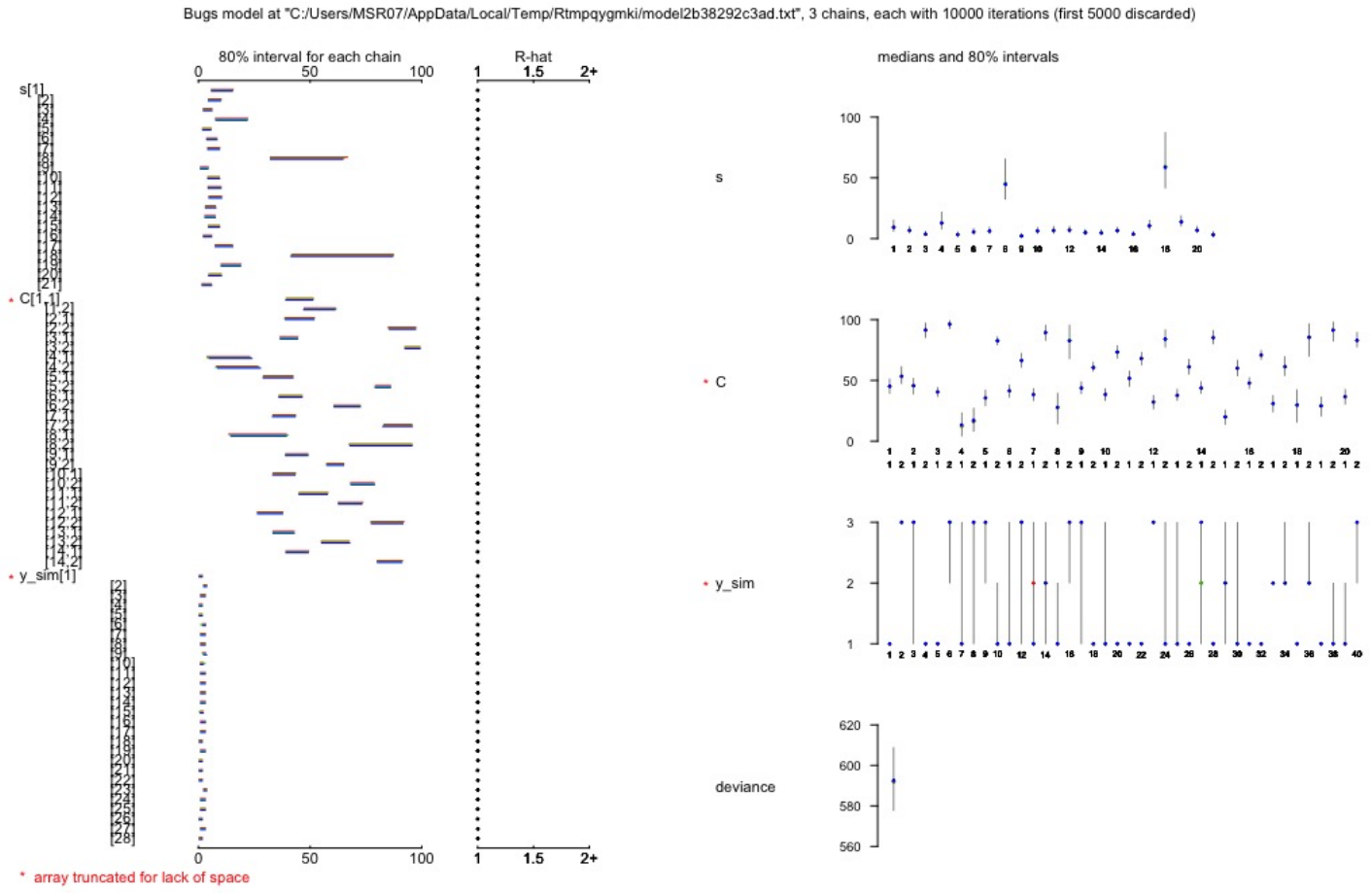
```
require(R2WinBUGS)
require(R2OpenBUGS)
vote_data <- read.csv('3playergames.csv')

y <- vote_data$vote
x <- vote_data$value
n <- nrow(vote_data)
player <- vote_data$person %>% factor %>% as.numeric
n_cut <- 2
n_player <- player %>% unique %>% length

order_logit <- function() {
 for (i in 1:n) {
   y[i] ~ dcat(P[i, ])
   y_sim[i] ~ dcat(P[i, ])
   P[i, 1] <- 1 - Q[i, 1]
   P[i, 2] <- Q[i, 1] - Q[i, 2]
   P[i, 3] <- Q[i, 2]
   for (k in 1:n_cut) {
     logit(Q[i, k]) <- Z[i, k]
     Z[i, k] <- (x[i] - C[player[i], k])/s[player[i]]
   }
 }
 for (j in 1:n_player) {
   C[j, 1] ~ dnorm(mu_c[1], tau_c[1])
   I(0, C[j, 2])
   C[j, 2] ~ dnorm(mu_c[2], tau_c[2])
   I(C[j, 1], 100)
   s[j] ~ dlnorm (mu_log_s, tau_log_s)
 }
 for (k in 1:n_cut){
   mu_c[k] ~ dnorm (0, 1.E-6)
   I(0, 100)
   tau_c[k] <- pow(sigma_c[k], -2)
   sigma_c[k] ~ dunif (0, 1000)
 }
 mu_log_s ~ dnorm (0, .00001)
 tau_log_s <- pow(sigma_log_s, -2)
 sigma_log_s ~ dunif (0, 1000)
}

data_vote <- list('y', 'x', 'n', 'n_cut', 'n_player', 'player')
C_inits <- array(dim= c(n_player,2))
for (j in 1:n_player){
 for (k in 1:2){
   C_inits[j,k] <- 25+5*k+rnorm(1)
 }
}
inits_vote <- function() {
 list(
   'y_sim'=y,
   "C"=C_inits,
   "s"=rep(10000, n_player),
```

```
   "mu_c"=c(20, 80),
   "sigma_c"=runif(n_cut),
   "mu_log_s"=rnorm(1),
   "sigma_log_s"=runif(1)
 )
}

# write.model(order_logit, 'oder_logit.bug')

params_vote <- c('s', 'C', 'y_sim')

vote_jags_model <- bugs(data_vote, inits_vote, params_vote, order_logit,
                        n.chains = 3, n.iter = 10000, debug=TRUE)
```

Output

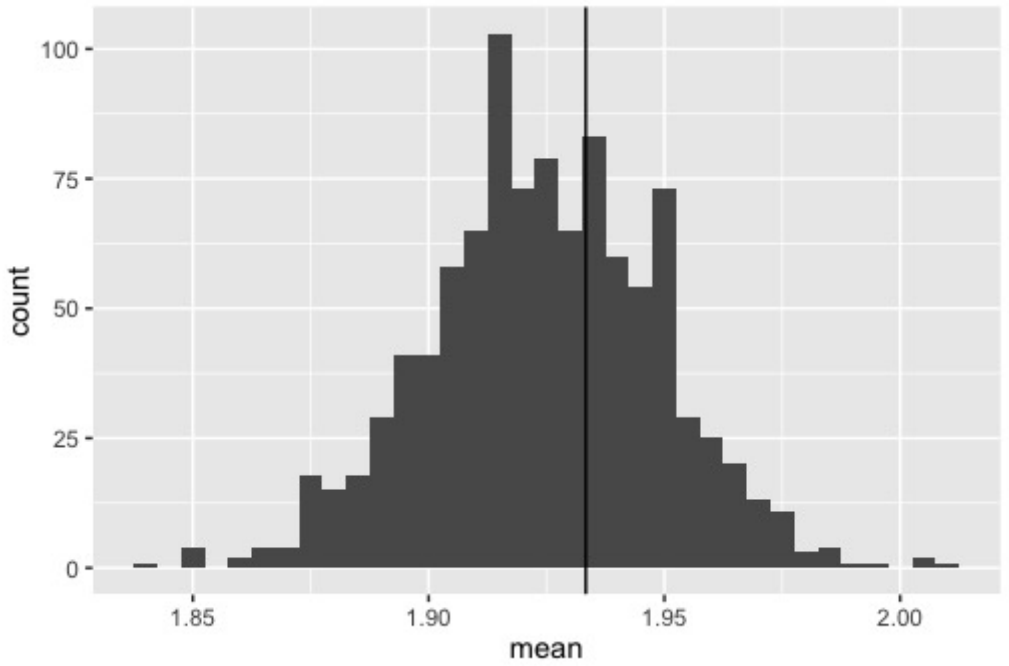
```
                mean          sd        2.5%       25%      50%       75%      97.5%
  Rhat n.eff
s[1]        10.181305  4.36542013   4.6929750   7.30200   9.2710  11.95250  21.490250 1.
001117  9200
s[2]         7.067415  2.25927448   3.6509500   5.45875   6.7190   8.31700  12.480250 1.
001083 11000
s[3]         3.870234  1.56559758   1.7059500   2.74400   3.5560   4.68200   7.735075 1.
001350  4400
s[4]        14.580618 10.84820575   6.0398749   9.82000  12.7550  16.79000  31.430500 1.
002030 15000
s[5]         3.546696  1.54022281   1.4149750   2.44500   3.2475   4.33000   7.323100 1.
001307  4900
...
s[15]        6.792026  1.99155116   3.7109750   5.39100   6.5240   7.86525  11.470000 1.
000927 15000
s[16]        3.855170  1.49329858   1.7520000   2.78900   3.5950   4.59300   7.493000 1.
000906 15000
s[17]       11.084995  3.28194063   6.4710000   8.81475  10.5100  12.72000  19.190250 1.
000949 15000
s[18]       62.342048 19.45617618  34.9800000  48.81750  58.9400  71.86000 110.200000 1.
001085 11000
s[19]       14.264241  3.47929161   8.8369250  11.77000  13.8000  16.27000  22.330000 1.
000941 15000
s[20]        7.201995  2.24216548   3.8449500   5.61875   6.8525   8.40350  12.570250 1.
000968 15000
s[21]        3.492759  1.71594086   1.1250000   2.26400   3.1640   4.35025   7.808075 1.
000949 15000
C[1,1]      45.271927  4.80034424  35.7397500  42.18000  45.2500  48.31000  54.830250 1.
001363  4300
C[1,2]      54.041903  5.78309925  44.3697500  50.16000  53.4100  57.28000  67.110750 1.
001682  2600
C[2,1]      45.449044  5.12170905  34.8597500  42.12000  45.6700  49.01000  54.920000 1.
001453  3600
C[2,2]      91.258556  4.53582565  81.9000000  88.20000  91.4500  94.60000  99.070000 1.
000999 15000
C[3,1]      40.511111  3.09110271  34.1797500  38.58000  40.5500  42.53000  46.460000 1.
000947 15000
C[3,2]      95.947665  2.52574230  90.5500000  94.26000  96.1900  97.93000  99.760000 1.
001035 15000
...
C[19,1]     28.743148  6.32876678  15.5394997  24.76000  29.1200  33.05000  40.160000 1.
001463 15000
C[19,2]     90.671262  6.05033463  77.5297500  86.58750  91.4000  95.56000  99.540000 1.
001078 11000
C[20,1]     36.559909  4.84658018  26.7200000  33.41000  36.5800  39.72000  46.180250 1.
000906 15000
C[20,2]     83.278933  4.80460593  74.3700000  80.04000  83.0200  86.26000  93.710500 1.
000957 15000
C[21,1]     16.444516  4.31706757   7.5599250  13.62000  16.7000  19.51000  24.170000 1.
001025 15000
C[21,2]     93.176995  3.02309148  87.1300000  91.21000  93.0900  95.23000  99.010000 1.
001281  5200
y_sim[1]     1.032867  0.22990572   1.0000000   1.00000   1.0000   1.00000   1.000000 1.
```

```
001819 15000
y_sim[2]     2.964400  0.24070003   3.0000000   3.00000   3.0000   3.00000   3.000000 1.
000916 15000
y_sim[3]     2.539000  0.75479599   1.0000000   2.00000   3.0000   3.00000   3.000000 1.
001106  9600
y_sim[4]     1.055133  0.29860720   1.0000000   1.00000   1.0000   1.00000   2.000000 1.
001231 12000
y_sim[5]     1.084333  0.36354280   1.0000000   1.00000   1.0000   1.00000   3.000000 1.
001012 15000
...
y_sim[625]   1.032200  0.17653679   1.0000000   1.00000   1.0000   1.00000   2.000000 1.
001196 12000
y_sim[626]   1.992067  0.09020867   2.0000000   2.00000   2.0000   2.00000   2.000000 1.
000901 15000
y_sim[627]   2.032333  0.17726623   2.0000000   2.00000   2.0000   2.00000   3.000000 1.
000981 15000
y_sim[628]   2.426333  0.49455993   2.0000000   2.00000   2.0000   3.00000   3.000000 1.
000956 15000
y_sim[629]   2.068333  0.25285335   2.0000000   2.00000   2.0000   2.00000   3.000000 1.
000994 15000
y_sim[630]   1.099000  0.29867197   1.0000000   1.00000   1.0000   1.00000   2.000000 1.
000931 15000
deviance   592.852240 12.26409560 570.8000000 584.30000 592.2000 600.70000 618.800000 1.
001105  9700
```

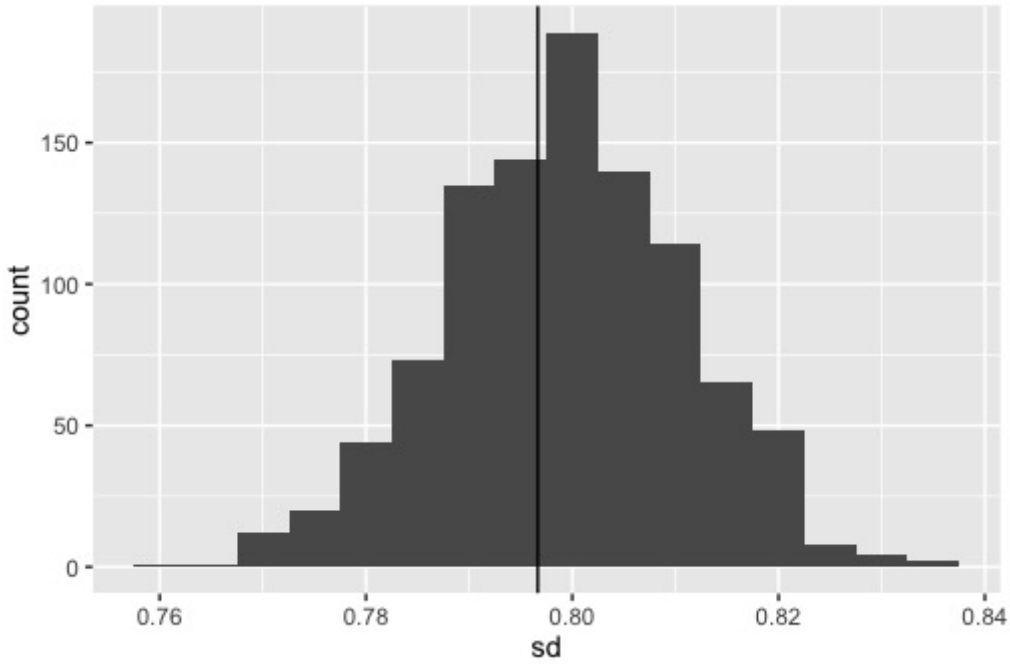


Plot

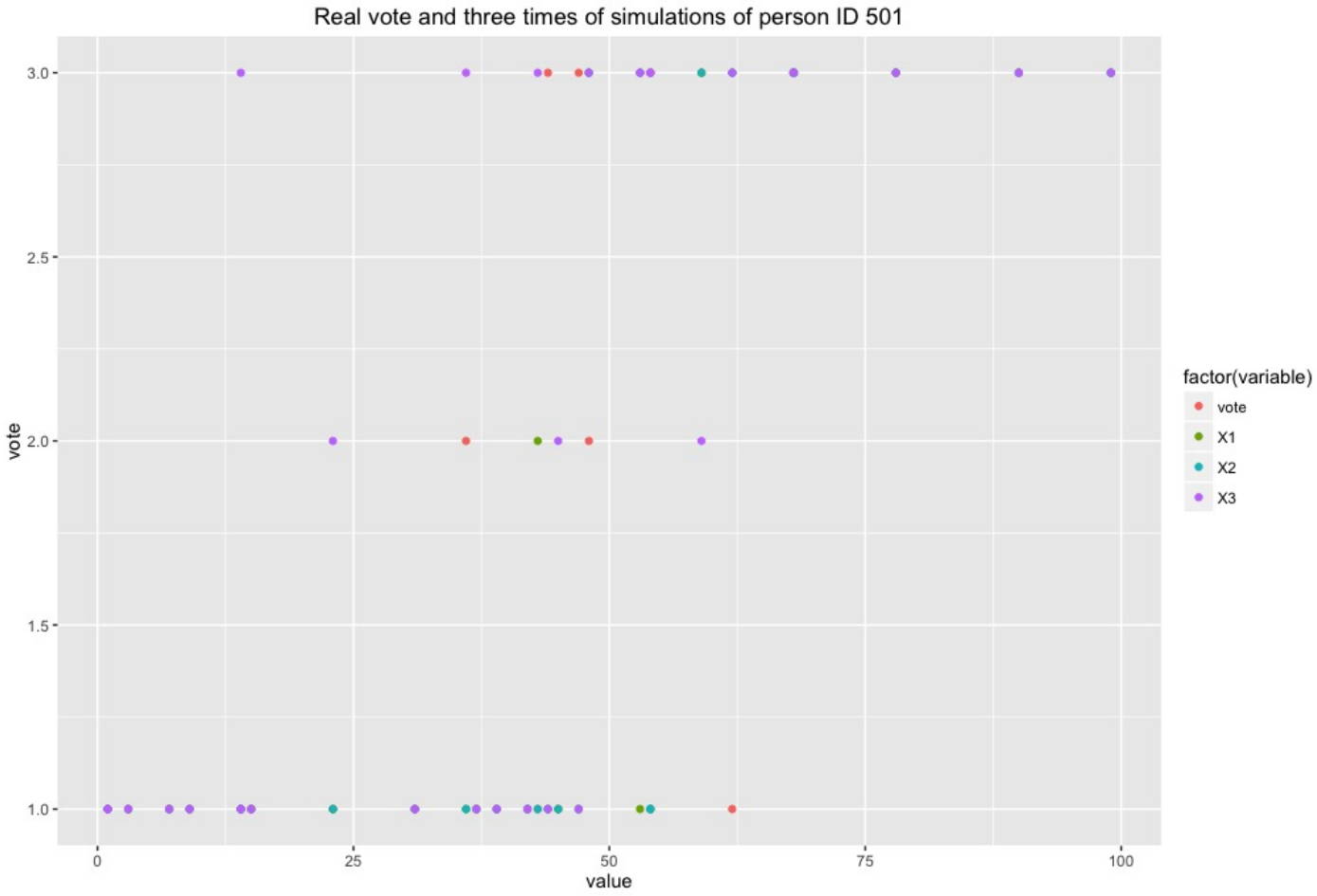
```
vote_data_plot <- data.frame(vote_data$person, vote_data$value, vote_data$vote, t(vote_j
ags_model$sims.list$y_sim[sample(1:15000, 1000), ]))
colnames(vote_data_plot)[1:3] <- c('person', 'value', 'vote')
vote_data_hist <- data.frame(mean = unlist(vote_data_plot[, -c(1:3)] %>% lapply(., mea
n)),
                             sd = unlist(vote_data_plot[, -c(1:3)] %>% lapply(., sd)))
ggplot(vote_data_hist, aes(x = mean)) + geom_histogram(binwidth = .005) + geom_vline(xin
tercept = mean(vote_data_plot$vote))
```




```
ggplot(vote_data_hist, aes(x = sd)) + geom_histogram(binwidth = .005) + geom_vline(xinte
rcept = sd(vote_data_plot$vote))
```



```
data_plot <- melt(vote_data_plot[, c(1:6)], id.vars = c("person", "value"))
colnames(data_plot)[4] <- "vote"
ggplot(data_plot[data_plot$person==501,], aes(x= value, y= vote, color = factor(variabl
e))) + geom_point() + ggtitle("Actual vote and three times of simulations of person ID 5
01")
```



Part B

How might you expand the model to fix the problems you have found?

I use openBUGS to fit the ordered logistic regression model. The histgram plots show the predictive checks for the mean and sd of the vote. The vertical bars indicate the mean and sd of the actual data. The scatter plot shows the actual vote and the first three times of simulations of the model. Based on these plots, this model is fitted reasonable well of the actual data. In order to improve the result, I think we can add more variables into the model, like round, or increase the number of simulations.